Apple Assembly Line

$1.50

Volume 3 -- Issue 4 January, 1983

In This Issue...

New Versions

Bob Matzinger has adapted the S-C Macro Assembler for the Basis 108. His revision uses the full capabilities of the Basis keyboard, and operates in 80-column mode. Bob also added some new Escape functions:

Escape A - Turns on automatic line numbering
Escape M - Turns off " " "
Escape C - Catalog the current disk
Escape S - Automatic SAVE to the file named in first line of program

If you have the Basis 108, and already own the S-C Macro Assembler, I'll send you a copy of the Basis version for $20.

We are currently working on an Apple /// version of the S-C Macro Assembler. Whew! It's no easy task! Do any of you use the Apple ///? My hat is off to you.

The Apple II-E should be out by the time you are reading this. We understand that there are changes in the ROM and DOS, which will probably affect our programs. If so, we'll let you know what the details are next month.

Super Scroller.....Jeffrey Scott
East Norwalk, CT

I am a manager of a software department in a company that makes computerized money counting equipment (6502 based). We have two programming departments: one which is called "applications" (Pascal and BASIC only) and another called "software engineering" where we use assembly language.

We use the S-C Macro Assembler after having sampled all others. And in fact, with my Apple II, 5 Mbyte hard drive, and 3.6 MHz "Number Nine 6502" plug-in board, I can assemble a 300-page source program in about 2.5 minutes!

I love the Apple II, but I don't like being tied to an operating system that I didn't write myself. I use RWTS, but for the rest I use my own code.

I remember one day trying to output to the screen while receiving at 2400 baud. The Apple monitor's scroll was so slow that I lost the first few characters from the front of every line. While writing my own substitute scroll routine, the idea was born that the absolute fastest scroll would be straight in-line code: one "LDA $xxxx...STA $xxxx" pair for each byte on the screen.

Just for fun, I wrote the following program, which generates the 960 LDA-STA pairs to scroll the whole screen! The generator program is only 145 bytes long, but it "writes" a program 5521 bytes long!

This "Super Scroller" is not for everyone...it requires a spare 5521 bytes ($1591) of memory somewhere. If you do, you need only equate "PGM.START.IN.RAM" to your available area, call "PGM.TO.WRITE.SCROLLING.PGM", and then you can call the Super Scroller at "PGM.START.IN.RAM whenever you need it.

Since the scroller can be generated whenever it is needed, it can be part of an overlay environment. You only need a 5.5K buffer available at the right times. At other times the same memory can be used other ways.

To illustrate the speediness of Super Scroller, I wrote a memory dump whose output is the same as the Apple monitor memory dump. It is set up to display from $0000 through $BFFF. With Super Scroller, it takes only about 51 seconds; without, it takes 2 minutes 57 seconds (over three times longer!).

Someone might object that I did not clear the bottom line after scrolling up. I elected to just write a fresh bottom line, and clear to the end of line after the last new character is written.

S-C Macro Assembler (the best there is!).....$80.00
Upgrade from Version 4.0 to MACRO.....$27.50
Source code of Version 4.0 on disk.....$95.00
Fully commented, easy to understand and modify to your own tastes.
S-C Macro Assembler /// (coming soon!).....$???.00

Applesoft Source Code on Disk.....$50.00
Very heavily commented. Requires Applesoft and S-C Assembler.

ES-CAPE: Extended S-C Applesoft Program Editor.....$60.00

AAL Quarterly Disks.....each $15.00
Each disk contains all the source code from three issues of "Apple Assembly Line", to save you lots of typing and testing time.
QD#1: Oct-Dec 1980 QD#2: Jan-Mar 1981 QD#3: Apr-Jun 1981
QD#4: Jul-Sep 1981 QD#5: Oct-Dec 1981 QD#6: Jan-Mar 1982
QD#7: Apr-Jun 1982 QD#8: Jul-Sep 1982 QD#9: Oct-Dec 1982

Double Precision Floating Point for Applesoft.....$50.00
Provides 21-digit precision for Applesoft programs.
Includes sample Applesoft subroutines for standard math functions.

FLASH! Integer BASIC Compiler (Laumer Research)..... $79.00
Source Code for FLASH! Runtime Package.....$39.00

Super Disk Copy III (Sensible Software).....(reg. $30.00) $27.00
Amper-Magic (Anthro-Digital).....(reg. $75.00) $67.50
Amper-Magic Volume 2 (Anthro-Digital).....(reg. $35.00) $30.00
Quick-Trace (Anthro-Digital).....(reg. $50.00) $45.00
Cross-Reference and Dis-Assembler (Rak-Ware).....$45.00
Apple White Line Trace (Lone Star Industrial Computing).....$50.00
(A unique learning tool)

Blank Diskettes (with hub rings).....package of 20 for $50.00
Small 3-ring binder with 10 vinyl disk pages and disks.....$36.00
Vinyl disk pages, 6"x8.5", hold one disk each.....10 for $4.50
Reload your own NEC PC-8023 ribbon cartridges.....each ribbon $5.00
Reload your own NEC Spinwriter Multi-Strike Film cartridges....each $2.50
Diskette Mailing Protectors.....10-99: 40 cents each
100 or more: 25 cents each

Ashby Shift-Key Mod.....$15.00
Lower-Case Display Encoder ROM.....$25.00
Only Revision level 7 or later Apples.

Books, Books, Books.....compare our discount prices!

"Enhancing Your Apple II, vol. 1", Lancaster.....	($15.95)	$15.00
"Incredible Secret Money Machine", Lancaster.....	($7.95)	$7.50
"Micro Cookbook, vol. 1", Lancaster.....	($15.95)	$15.00
"Beneath Apple DOS", Worth & Lechner.....	($19.95)	$18.00
"Bag of Tricks", Worth & Lechner, with diskette.....	($39.95)	$36.00
"Apple Graphics & Arcade Game Design", Stanton.....	($19.95)	$18.00
"Assembly Lines: The Book", Roger Wagner.....	($19.95)	$18.00
"What's Where in the Apple", Second Edition.....	($24.95)	$23.00
"What's Where Guide" (updates first edition).....	($9.95)	$9.00
"6502 Assembly Language Programming", Leventhal.....	($16.99)	$16.00
"6502 Subroutines", Leventhal.....	($12.99)	$12.00
"MICRO on the Apple--1", includes diskette.....	($24.95)	$23.00
"MICRO on the Apple--2", includes diskette.....	($24.95)	$23.00
"MICRO on the Apple--3", includes diskette.....	($24.95)	$23.00

Add $1 per book for US postage. Foreign orders add postage needed.

*** S-C SOFTWARE, P. O. BOX 280300, Dallas, TX 75228 ***
*** (214) 324-2050 ***
*** We take Master Charge, VISA and American Express ***

```
                    1000 *SAVE SUPER SCROLL GENERATOR
                    1010 *--------------------------------
                    1020 *
                    1030 *    APPLE SUPER SCROLLER
                    1040 *
                    1050 *--------------------------------
                    1060 *      PROGRAM TO CREATE A FAST SCROLLER
                    1070 *
                    1080 *      CREATES AN ALL "IN-LINE" SCROLL ROUTINE
                    1090 *      WHICH MAY BE CALLED AS A SUBROUTINE.
                    1100 *
                    1110 *      WILL SCROLL LINES 2-24 UP TO LINES 1-23
                    1120 *      IN ONLY 7.6 MILLISECONDS.
                    1130 *
                    1140 *      BOTTOM LINE IS LEFT UNCHANGED; YOU MAY
                    1150 *      WISH TO ADD MORE CODE TO BLANK BOTTOM LINE.
                    1160 *--------------------------------
                    1170
4000-               1180 PGM.START.IN.RAM      .EQ $4000
0002-               1190 PROGRAM               .EQ $02 - $03
0004-               1200 UPPER.LINE            .EQ $04 - $05
0006-               1210 LOWER.LINE            .EQ $06 - $07
                    1220 *--------------------------------
                    1230        .MA SCRN
                    1240        .DA ]1,]1+$80,]1+$100,]1+$180
                    1250        .DA ]1+$200,]1+$280,]1+$300,]1+$380
                    1260        .EM
                    1270 *--------------------------------
                    1280 APPLE.SCREEN.ADDRESSES
0800-               1290        >SCRN $400     LINES 1-8
0800- 00 04 80
0803- 04 00 05
0806- 80 05         0000>        .DA $400,$400+$80,$400+$100,$400+$180
0808- 00 06 80
080B- 06 00 07
080E- 80 07         0000>        .DA $400+$200,$400+$280,$400+$300,$400+$380
0810-               1300        >SCRN $428     LINES 9-16
0810- 28 04 A8
0813- 04 28 05
0816- A8 05         0000>        .DA $428,$428+$80,$428+$100,$428+$180
0818- 28 06 A8
081B- 06 28 07
081E- A8 07         0000>        .DA $428+$200,$428+$280,$428+$300,$428+$380
0820-               1310        >SCRN $450     LINES 17-24
0820- 50 04 D0
0823- 04 50 05
0826- D0 05         0000>        .DA $450,$450+$80,$450+$100,$450+$180
0828- 50 06 D0
082B- 06 50 07
082E- D0 07         0000>        .DA $450+$200,$450+$280,$450+$300,$450+$380
                    1320 *--------------------------------
                    1330 PGM.TO.WRITE.SCROLLING.PGM
                    1340
0830- A9 00         1350        LDA #PGM.START.IN.RAM
0832- 85 02         1360        STA PROGRAM
0834- A9 40         1370        LDA /PGM.START.IN.RAM
0836- 85 03         1380        STA PROGRAM+1
                    1390 *--------------------------------
0838- A2 00         1400        LDX #0        FOR LINE = 1 TO 23
083A- BD 00 08      1410 .1     LDA APPLE.SCREEN.ADDRESSES,X
083D- 85 04         1420        STA UPPER.LINE
```

```
083F- BD 01 08 1430         LDA APPLE.SCREEN.ADDRESSES+1,X
0842- 85 05    1440         STA UPPER.LINE+1
               1450
0844- BD 02 08 1460         LDA APPLE.SCREEN.ADDRESSES+2,X
0847- 85 06    1470         STA LOWER.LINE
0849- BD 03 08 1480         LDA APPLE.SCREEN.ADDRESSES+3,X
084C- 85 07    1490         STA LOWER.LINE+1
               1500
084E- 8A       1510         TXA            SAVE LINE #
084F- 48       1520         PHA
               1530 *-------------------------------
0850- A2 28    1540         LDX #40        FOR CHAR = 1 TO 40
0852- A0 00    1550 .2      LDY #0
0854- A9 AD    1560         LDA #$AD         "LDA ABSOLUTE"
0856- 91 02    1570         STA (PROGRAM),Y
0858- C8       1580         INY
0859- A5 06    1590         LDA LOWER.LINE
085B- 91 02    1600         STA (PROGRAM),Y
085D- C8       1610         INY
085E- A5 07    1620         LDA LOWER.LINE+1
0860- 91 02    1630         STA (PROGRAM),Y
0862- C8       1640         INY
0863- A9 8D    1650         LDA #$8D         "STA ABSOLUTE"
0865- 91 02    1660         STA (PROGRAM),Y
0867- C8       1670         INY
0868- A5 04    1680         LDA UPPER.LINE
086A- 91 02    1690         STA (PROGRAM),Y
086C- C8       1700         INY
086D- A5 05    1710         LDA UPPER.LINE+1
086F- 91 02    1720         STA (PROGRAM),Y
               1730 *-------------------------------
0871- 98       1740         TYA            UPDATE PROGRAM POINTER
0872- 38       1750         SEC
0873- 65 02    1760         ADC PROGRAM
0875- 85 02    1770         STA PROGRAM
0877- 90 02    1780         BCC .3
0879- E6 03    1790         INC PROGRAM+1
087B- E6 04    1800 .3      INC UPPER.LINE     NEXT CHAR POSITION
087D- E6 06    1810         INC LOWER.LINE
087F- CA       1820         DEX
0880- D0 D0    1830         BNE .2
               1840 *-------------------------------
0882- 68       1850         PLA
0883- AA       1860         TAX
0884- E8       1870         INX            NEXT LINE
0885- E8       1880         INX
0886- E0 2E    1890         CPX #2*23
0888- D0 B0    1900         BNE .1
               1910 *-------------------------------
088A- A0 00    1920         LDY #0
088C- A9 60    1930         LDA #$60         "RTS"
088E- 91 02    1940         STA (PROGRAM),Y
0890- 60       1950         RTS
               1960 *-------------------------------
               1970 * A FAST MEMORY DUMP!!
               1980 *-------------------------------
0008-          1990 MEML                   .EQ $8
0009-          2000 MEMH                   .EQ $9
07D0-          2010 SCREEN.WRITE.LINE      .EQ $7D0
               2020 *-------------------------------
```

&Amper-Magic™

MACHINE LANGUAGE SPEED WHERE IT COUNTS...

IN YOUR PROGRAM!

For the first time, Amper-Magic makes it easy for people who don't know machine language to use its power! Now you can attach slick, finished machine language routines to your Applesoft programs in seconds! And interface them by name, not by address!

You simply give each routine a name of your choice, perform the append procedure once at about 15 seconds per routine, and the machine language becomes a permanent part of your BASIC program. (Of course, you can remove it if you want to.)

Up to 255 relocatable machine language routines can be attached to a BASIC program and then called by name. We supply some 20 routines on this disk. More can be entered from magazines. And more library disks are in the works.

These routines and more can be attached and accessed easily. For example, to allow the typing of commas and colons in a response (not normally allowed in Applesoft), you just attach the Input Anything routine and put this line in your program:

```
xxx  PRINT "PLEASE ENTER THE DATE."; : & INPUT,DATE$
```

Some routines on this disk are:

Binary file info
Delete array
Disassemble memory
Dump variables
Find substring
Get 2-byte values
Gosub to variable
Goto to variable
Hex memory dump
Input anything
Move memory
Multiple poke decimal
Multiple poke hex
Print w/o word break
Restore special data
Speed up Applesoft
Speed restore
Store 2-byte values
Swap variables

&-MAGIC makes it Easy to be Fast & Flexible!

PRICE: $75

Anthro - Digital Software
P.O. Box 1385
Pittsfield, MA 01202

The People - Computers Connection

```
                    2030 START.DEMO
0891- 20 30 08      2040         JSR PGM.TO.WRITE.SCROLLING.PGM
                    2050 MEMDUMP
0894- A9 00         2060         LDA #0        DISPLAY FROM $0000 THRU $BFFF
0896- 85 08         2070         STA MEML
0898- 85 09         2080         STA MEMH
089A- A2 00         2090 .1      LDX #0        X = CHAR PNTR IN OUTPUT LINE
089C- A5 09         2100         LDA MEMH      DISPLAY ADDRESS
089E- 20 E1 08      2110         JSR DISPLAY.BYTE
08A1- A5 08         2120         LDA MEML
08A3- 20 E1 08      2130         JSR DISPLAY.BYTE
08A6- A9 AD         2140         LDA #$AD      "- "
08A8- 9D D0 07      2150         STA SCREEN.WRITE.LINE,X
08AB- E8            2160         INX
08AC- A9 A0         2170         LDA #$A0
08AE- 9D D0 07      2180         STA SCREEN.WRITE.LINE,X
08B1- E8            2190         INX
08B2- A0 00         2200         LDY #0
08B4- B1 08         2210 .2      LDA (MEML),Y       DISPLAY 8 BYTES
08B6- 20 E1 08      2220         JSR DISPLAY.BYTE
08B9- A9 A0         2230         LDA #$A0
08BB- 9D D0 07      2240         STA SCREEN.WRITE.LINE,X
08BE- E8            2250         INX
08BF- C8            2260         INY
08C0- C0 08         2270         CPY #8
08C2- D0 F0         2280         BNE .2
08C4- 9D D0 07      2290 .3      STA SCREEN.WRITE.LINE,X
08C7- E8            2300         INX
08C8- E0 28         2310         CPX #40       CLEAR TO END OF LINE
08CA- 90 F8         2320         BCC .3
                    2330 *-------------------------------
08CC- 20 00 40      2340         JSR PGM.START.IN.RAM
                    2350 *-------------------------------
08CF- A9 08         2360         LDA #8
08D1- 18            2370         CLC
08D2- 65 08         2380         ADC MEML
08D4- 85 08         2390         STA MEML
08D6- A5 09         2400         LDA MEMH
08D8- 69 00         2410         ADC #0
08DA- 85 09         2420         STA MEMH
08DC- C9 C0         2430 .4      CMP #$C0      STOP AT $BFFF
08DE- D0 BA         2440         BNE .1
08E0- 60            2450         RTS
                    2460 *-------------------------------
                    2470 DISPLAY.BYTE
08E1- 48            2480         PHA
08E2- 4A            2490         LSR
08E3- 4A            2500         LSR
08E4- 4A            2510         LSR
08E5- 4A            2520         LSR
08E6- 20 EC 08      2530         JSR DISPLAY.NYBBLE
08E9- 68            2540         PLA
08EA- 29 0F         2550         AND #$0F
                    2560 DISPLAY.NYBBLE
08EC- 09 B0         2570         ORA #$B0      MAKE HEX DIGIT
08EE- C9 BA         2580         CMP #$BA
08F0- 90 02         2590         BCC .1
08F2- 69 06         2600         ADC #6
08F4- 9D D0 07      2610 .1      STA SCREEN.WRITE.LINE,X
08F7- E8            2620         INX
08F8- 60            2630         RTS
```

Micro Cookbook, vol. 1 (Review).....Bill Morgan

Here are some more details about Don Lancaster's other new book, "Micro Cookbook, vol. 1 -- Fundamentals." As I said last month, the focus of the book is what to learn and how to learn it. He emphasizes "what actually gets used", rather than an exhaustive coverage of all possibilities.

The best quick description of the book is an excerpt from the Preface:

> Our aim is to show you how micros work, and how you can profit from and enjoy the micro revolution.
>
> We start with the power and the underlying idea behind all micros. From there we build up the framework for all the important micro concepts and terms. The micro-processor families are broken down into three simple and easily understood schools.
>
> Chapter Two starts with a set of rules for winning the micro game. These rules have been thoroughly tested in the real world and are not at all what you might expect. After that, we check into many of the resources that are available to you as a micro user. A survey of micro trainers is included.
>
> The Funny Numbers section (Chapter 3) shows you ways to use and understand the number systems involved in micros, particularly binary and hexadecimal. From there, we look at logic, both as hardware gates and as software commands.
>
> The fourth chapter is all about codes. The important codes that are covered include straight binary, 2's complement binary, ASCII, BCD, instruction codes, user port codes, and various serial data-transmission codes and standards. The 2's complement codings are presented in a new and understandable way.
>
> Chapter 5 tells us many things about memory. We go into electronic memory -- beginning with simple latches and progressing to clocked flip-flops. Mainstream microcomputer memory is attacked next -- from static RAMs up through dynamic RAM, ROM, PROM, EPROM, and EEPROM memories.

"Micro Cookbook -- Fundamentals" is just that: Fundamental. I am a complete novice on hardware. After reading Lancaster's book, I still can't design custom interfaces for my Apple, but I can now read the more technical books without getting totally lost. I have a better understanding of address decoding and of what the memory chips are really doing. The book is informative, enlightening, and entertaining. I recommend it.

This Cookbook is about 360 pages of text, plus appendices and index. There are many drawings and charts. List price is $15.95. We will be selling it for $15.00 + postage.

Funny Opcode Names in the 6801 Manual......Bob Sander-Cederlof

Paul Lundgren (of Microcomp, Inc. in Newtown, CT) brought some interesting facts to my attention today. When I implemented my 6801 Cross Assemblers, I used what was at the time the latest documentatin available. Paul had some printed two years later, and there were some differences.

For some reason, the Motorola 6801 Reference Manual changes the name of the ASL and ASLD opcodes to LSL and LSLD. There is no difference in operation, just a difference in spelling. The S-C Cross Assembler only recognizes the ASL and ASLD spellings. The opcode tables are near the end of the assembler, so you can easily find these entries to change them if you feel strongly about it.

The Motorola book also lists alias names for the BCC and BCS opcodes. In the 6801 (or other 68xx chips), carry clear means the last test was greater or equal, so the alias name is BHS (Branch if High or Same). Carry set means the test was smaller, so the alias is BLO. Note that the meaning of carry after a comparison in the 68xx chips is exactly the opposite of carry in the 6502!

Here are some macros to use for BHS and BLO:

```
.MA BHS
BCC ]1
.EM

.MA BLO
BCS ]1
.EM
```

Some assemblers for the 6502 have two alias opcodes for BCC and BCS. For example, LISA has BLT for BCC (Branch if Less Than), and BGE for BCS (Branch if Greater than or Equal). [I didn't do this in the S-C Assemblers because the meaning depends on whether the values tested are considered to be signed or unsigned.]

Here are two macros to implement BLT and BGE in the 6502 version of the S-C Macro Assembler:

```
.MA BLT
BCC ]1
.EM

.MA BGE
BCS ]1
.EM
```

An Addition to CATALOG ARRANGER.....Dave Barkovitch

I really like Bill Morgan's CATALOG ARRANGER, from the October issue of AAL. There is something I want to change, though.

When you move the cursor to the end of a long catalog, the cursor stays in the middle of the screen and the catalog scrolls up, until only the top half of the screen is filled. Here are some patches to make the cursor move down to the end, and keep 22 files on the screen:

```
2931        LDA NUMBER.OF.ELEMENTS
2932        SEC
2933        SBC #LINE.COUNT
2934        BPL .5
2935        LDA #ZERO
2936 .5     STA LAST.ELEMENT


3830        BPL .7

3841        BEQ .1
3842 .7     CMP LAST.ELEMENT
3843        BCC .1
3844        LDA LAST.ELEMENT

5991 LAST.ELEMENT .BS 1
```

And Another Change.....Bill Collins

CATALOG ARRANGER is a great utility. Here are a couple of things you might like to know:

1. Version 4.0 of the S-C Assembler will not accept division in the operand. If you have that version then change line 3820 to SBC #11.

2. If you have DOS relocated into a RAM card you need to add the following lines for bank switching purposes:

```
1165 MONREAD .EQ $C082
1167 DOSREAD .EQ $C083
```

Then add BIT MONREAD at these positions: Lines 1675, 3785, 3855, 3895, 4015 (".5" moved to this line), 4205 (".3" moved to this line), 4315, 4425, 4455 (".7" moved to this line).

And add BIT DOSREAD at these spots: Lines 1535-36, 1685-86, 3795-96, 3905-06, 3975-76, 4035-36, 4215-16, 4345-46, 4465-66, 4955-56.

Also, all DOS addresses must be moved up 16K (lines 1180-1310.) $Axxx addresses become $Exxx and $Bxxx become $Fxxx.

A Filename Editor for CATALOG ARRANGER.....Bill Morgan

Many thanks to all of you who have called and written to say how much you like the CATALOG ARRANGER. I'm glad to hear that others find it as useful as I do. Here's my favorite addition to the program, the ability to edit the filename in the cursor. Now you can alter a name by inserting or deleting characters, you can insert control characters, and you can place display titles in the catalog, using normal, inverse, flashing, or lower case text.

There are a couple of unique features in this editor. The cursor clearly indicates Insert, Overtype, or Override mode, and also shows whether the input will be Normal, Inverse, Flashing, or Lower Case. The display unambiguously shows all these types, plus Control. The price of all this clarity is three display lines for one text line, but that's no problem in a program like this. You can easily adapt the concepts shown here to edit any line of forty or fewer characters. The same principles also apply to longer lines, but the screen display would have to be handled carefully.

Installation

To add FILENAME EDITOR to CATALOG ARRANGER just type in S.FILENAME.EDITOR from this listing, and save it on the same disk with S.CATALOG.ARRANGER. Then LOAD S.CATALOG.ARRANGER and make the following changes and additions:

```
1030        .TF CATALOG.ARRANGER.NEW

1480 LINE.COUNT   .EQ 21

1915        CMP #$85    ^E
1920        BNE .1
1922        JSR RENAME.FILE
1924        JMP DISPLAY.AND.READ.KEY

5865        .IN S.FILENAME.EDITOR
```

Then SAVE the new S.CATALOG.ARRANGER and assemble it.

Operation

To rename a file, just use the arrow keys to move the cursor to the file you want, and type "CTRL-E" (for Edit). The name you selected will appear near the bottom of the screen, between square brackets. Any control characters in the name will have a bar above them. The caret below the first character of the name is the cursor. Any non-control characters you type will replace the characters on the screen. Control characters will have the effects shown in the command list below. Especially note that RETURN will enter the name in the lower buffer into the filename array, ESC will return you to the Arranger without altering the filename, and CTRL-R will restore the original filename.

One way to have fun with this program is to put dummy files in the catalog, for titles or just for decoration. First get into Applesoft and SAVE as many dummy programs (10 REM, for example) as you need. Then BRUN CATALOG ARRANGER, move the dummy programs to where you want them, and play with the names. If you start the new file name with six CTRL-H's and six spaces, it will blank out the "A 002 " before the name. You can use inverse, flashing or lower case text in titles. If you insert CTRL-M's (RETURNS) after a name there will be blank lines in the catalog. Play with it for a while, and let me know if you come up with any especially neat tricks.

Here are the commands:

<-- -- Left Arrow. Move the cursor left one position.
--> -- Right Arrow. Move the cursor right one position.
RETURN -- Enter. Enter the changed name into the upper display and return to arranging.
ESC -- Escape. Return to arranging, without entering the changed name.
^B -- Beginning. Move the cursor to the beginning of the line.
^D -- Delete. Delete one character at the cursor.
^E -- End. Move the cursor to the end of the name.
^F -- Find. Move the cursor to a particular character. Type "^FA" to move the cursor to the next "A" in the name. Type another "A" to move to the following "A", and so on. Any character other than the search key will be entered or executed.
^I -- Insert. Turn on Insert Mode. Following characters will be inserted to the left of the backslash cursor. Any control character turns Insert off.
^O -- Override. Insert the next character typed "as is". This allows you to insert control characters into a name.
^R -- Restore. Restore the name to its original condition, as it appears in the upper display.
^S -- Shift Mode. Cycle between Normal, Inverse, Flashing, and Lower Case entry. The cursor changes to show the current mode.
^Z -- Zap. Remove all characters from the cursor to the end of the name.

How it All Works

When you type CTRL-E to enter the editor, line 1090 transfers the filename into an edit buffer located in the screen memory at $757-$774. The main loop of the editor is lines 1190-1320. All through the editor the Y-register is the cursor position in the line. The routine DISPLAY.EDIT.BUFFER places the brackets before and after the name, puts bars over any control characters, displays the cursor, and gets the next keystroke. The main loop then checks whether that key was a control.

```
                    1590 E.OVERRIDE
0BA0- 68            1600         PLA
0BA1- 68            1610         PLA
0BA2- A9 A2         1620         LDA #$A2              SET CURSOR
0BA4- 99 D7 07      1630         STA CURSOR.LINE,Y     TO "
0BA7- 20 A1 0C      1640         JSR GETKEY
0BAA- 4C 81 0B      1650         JMP INSERT.CHARACTER
                    1660 *-------------------------------
                    1670 E.LEFT.ARROW
0BAD- 88            1680         DEY           MOVE CURSOR LEFT
0BAE- 10 01         1690         BPL .1        IF IT WENT NEGATIVE
0BB0- C8            1700         INY           RESTORE IT TO 0
0BB1- 60            1710 .1      RTS
                    1720 *-------------------------------
                    1730 E.RIGHT.ARROW
0BB2- C0 1D         1740         CPY #29       AT END YET?
0BB4- B0 01         1750         BCS .1        YES, IGNORE
0BB6- C8            1760         INY           NO, MOVE CURSOR RIGHT
0BB7- 60            1770 .1      RTS
                    1780 *-------------------------------
                    1790 E.INSERT
0BB8- A9 FF         1800         LDA #$FF         TURN INSERT ON
0BBA- 8D F7 0C      1810         STA INPUT.MODE
0BBD- A9 DC         1820         LDA #$DC
0BBF- 8D FB 0C      1830         STA CURSOR
0BC2- 60            1840         RTS
                    1850 *-------------------------------
                    1860 E.DELETE
0BC3- 98            1870         TYA                   SET X TO
0BC4- AA            1880         TAX                   CURSOR
0BC5- E0 1D         1890 .1      CPX #29               AT END?
0BC7- F0 09         1900         BEQ .2                BRANCH IF SO
0BC9- BD 58 07      1910         LDA EDIT.BUFFER+1,X
0BCC- 9D 57 07      1920         STA EDIT.BUFFER,X MOVE ONE CHAR
0BCF- E8            1930         INX                   NEXT
0BD0- 90 F3         1940         BCC .1             ...ALWAYS
0BD2- A9 A0         1950 .2      LDA #SPACE            PUT SPACE
0BD4- 9D 57 07      1960         STA EDIT.BUFFER,X ON END
0BD7- 60            1970         RTS
                    1980 *-------------------------------
                    1990 E.BEGINNING
0BD8- A0 00         2000         LDY #ZERO             ZERO CURSOR
0BDA- 60            2010         RTS
                    2020 *-------------------------------
                    2030 E.END
0BDB- A0 1D         2040         LDY #29               START AT END OF BUFFER
0BDD- B9 57 07      2050 .1      LDA EDIT.BUFFER,Y
0BE0- C9 A0         2060         CMP #SPACE            SPACE?
0BE2- D0 03         2070         BNE .2                NO, WE'RE AT END OF NAME
0BE4- 88            2080         DEY                   YES, MOVE LEFT
0BE5- 10 F6         2090         BPL .1                AND TRY AGAIN
0BE7- C0 1D         2100 .2      CPY #29               STILL AT END OF BUFFER?
0BE9- F0 01         2110         BEQ .3                YES, STAY THERE
0BEB- C8            2120         INY                   NO, RIGHT ONE SPACE
0BEC- 60            2130 .3      RTS
                    2140 *-------------------------------
                    2150 E.RESTORE
0BED- 68            2160         PLA                   POP A RETURN
0BEE- 68            2170         PLA                   ADDRESS AND
0BEF- 4C 43 0B      2180         JMP RENAME.FILE       START OVER
```

If the keypress was not a control character, it is passed to the input section (lines 1340-1570), where the character is masked according to the current MASK.MODE (Normal, Inverse, Flashing, or Lower-case) and either inserted or just placed in the line. The program then jumps back to E.START to redisplay the buffer and get the next key.

If you do enter a control character, the program JSR's to the SEARCH.AND.PERFORM routine at lines 3250-3390 (taken straight from Bob's article in the August '82 AAL.) Here we look up the command key in the table at lines 3420-3550 and do a PHA, PHA, RTS type branch to the appropriate command handler, or to the monitor's BELL, if the command didn't match anything in the table.

Almost all of the command handlers end with an RTS that returns control to line 1320. The exceptions are OVERRIDE (lines 1590-1650) and RESTORE (lines 2150-2180), since they exit through internal JMP's, and RETURN/ESC (lines 2660-2720), since those return to the main program. Another oddity is the FIND routine (lines 2420-2640), since it has two exits. Line 2640 returns to line 1320 through the BELL routine. Lines 2590-2620 are needed to process a keystroke that is not a repetition of the search key.

```
                    1010 *--------------------------------
0034-               1020 MON.YSAVE     .EQ $34
06D7-               1030 CONTROL.LINE  .EQ $6D7
0757-               1040 EDIT.BUFFER   .EQ $757
07D7-               1050 CURSOR.LINE   .EQ $7D7
FF3A-               1060 MON.BELL      .EQ $FF3A
                    1070 *--------------------------------
                    1080 RENAME.FILE
0B43- 20 48 0C      1090         JSR MOVE.FILE.INTO.BUFFER
0B46- A0 FF         1100         LDY #$FF
0B48- 8C F9 0C      1110         STY MASK.ONE       INITIALIZE
0B4B- C8            1120         INY
0B4C- 8C F7 0C      1130         STY INPUT.MODE     VARIABLES
0B4F- 8C FA 0C      1140         STY MASK.TWO
0B52- 8C F8 0C      1150         STY MASK.MODE
0B55- A9 DE         1160         LDA #$DE
0B57- 8D FB 0C      1170         STA CURSOR
                    1180 *--------------------------------
                    1190 E.START
0B5A- 20 6E 0C      1200         JSR DISPLAY.EDIT.BUFFER  UPDATE DISPLAY
                    1210 *                                AND GET KEYSTROKE
                    1220
                    1230 REENTRY
0B5D- C9 A0         1240         CMP #$A0                 CONTROL?
0B5F- B0 12         1250         BCS E.INPUT              NO, INPUT IT
0B61- A9 00         1260         LDA #ZERO                YES,
0B63- 8D F7 0C      1270         STA INPUT.MODE           TURN OFF INSERT
0B66- A9 DE         1280         LDA #$DE                 ^
0B68- 8D FB 0C      1290         STA CURSOR
0B6B- A2 00         1300         LDX #ZERO
0B6D- 20 B0 0C      1310         JSR SEARCH.AND.PERFORM   GO DO SOMETHING
0B70- 4C 5A 0B      1320         JMP E.START
                    1330 *--------------------------------
                    1340 E.INPUT
0B73- 2D F9 0C      1350         AND MASK.ONE          CONDITION
0B76- 0D FA 0C      1360         ORA MASK.TWO          CHARACTER
0B79- 8D F5 0C      1370         STA CURRENT.CHAR
0B7C- 2C F7 0C      1380         BIT INPUT.MODE        INSERT OR OVERTYPE?
0B7F- 10 11         1390         BPL PLACE.CHARACTER
                    1400
                    1410 INSERT.CHARACTER
0B81- 84 34         1420         STY MON.YSAVE         SAVE CURSOR
0B83- A2 1D         1430         LDX #29               START AT END OF BUFFER
0B85- E4 34         1440 .1      CPX MON.YSAVE         TO CURSOR YET?
0B87- F0 09         1450         BEQ PLACE.CHARACTER   YES
0B89- BD 56 07      1460         LDA EDIT.BUFFER-1,X   NO, MOVE CHAR UP
0B8C- 9D 57 07      1470         STA EDIT.BUFFER,X     TO MAKE HOLE
0B8F- CA            1480         DEX                   NEXT CHAR
0B90- 10 F3         1490         BPL .1             ...ALWAYS
                    1500
                    1510 PLACE.CHARACTER
0B92- AD F5 0C      1520         LDA CURRENT.CHAR
0B95- 99 57 07      1530         STA EDIT.BUFFER,Y
0B98- C0 1D         1540         CPY #29               END OF BUFFER?
0B9A- B0 01         1550         BCS .1                YES, RETURN
0B9C- C8            1560         INY                   NO, MOVE CURSOR
0B9D- 4C 5A 0B      1570 .1      JMP E.START
                    1580 *--------------------------------
```

S-C Macro Cross Assemblers

The high cost of dedicated microprocessor development systems has forced many technical people to look for alternate methods to develop programs for the various popular microprocessors. Combining the versatile Apple II with the S-C Macro Assembler provides a cost effective and powerful development system. Hobbyists and engineers alike will find the friendly combination the easiest and best way to extend their skills to other microprocessors.

The S-C Macro Cross Assemblers are all identical in operation to the S-C Macro Assembler; only the language assembled is different. They are sold as upgrade packages to the S-C Macro Assembler. The S-C Macro Assembler, complete with 100-page reference manual, costs $80; once you have it, you may add as many Cross Assemblers as you wish at a nominal price. The following S-C Macro Cross Assembler versions are now available, or soon will be:

Motorola:	6800/6801/6802	now	$32.50
	6805	now	$32.50
	6809	now	$32.50
	68000	now	$50
Intel:	8048	now	$32.50
	8051	soon	$32.50
	8085	soon	$32.50
Zilog:	Z-80	now	$32.50
RCA:	1802/1805	soon	$32.50
Rockwell:	65C02	now	$20

The S-C Macro Assembler family is well known for its ease-of-use and powerful features. Thousands of users in over 30 countries and in every type of industry attest to its speed, dependablility, and user-friendliness. There are 20 assembler directives to provide powerful macros, conditional assembly, and flexible data generation. INCLUDE and TARGET FILE capabilities allow source programs to be as large as your disk space. The integrated, co-resident source program editor provides global search and replace, move, and edit. The EDIT command has 15 sub-commands combined with global selection.

Each S-C Assembler diskette contains two complete ready-to-run assemblers: one is for execution in the mother-board RAM; the other executes in a 16K RAM Card. The HELLO program offers menu selection to load the version you desire. The disks may be copied using any standard Apple disk copy program, and copies of the assembler may be BSAVEd on your working disks.

S-C Software Corporation has frequently been commended for outstanding support: competent telephone help, a monthly (by subscription) newsletter, continuing enhancements, and excellent upgrade policies.

S-C Software Corporation (214) 324-2050
P.O. Box 280300, Dallas, Texas, 75228

```
                    2190 *--------------------------------
                    2200 E.SET.MODE
0BF2- EE F8 0C      2210         INC MASK.MODE          NEXT MODE
0BF5- AD F8 0C      2220         LDA MASK.MODE          IF MODE = 4
0BF8- 29 03         2230         AND #3                 MAKE IT ZERO
0BFA- 8D F8 0C      2240         STA MASK.MODE
0BFD- AA            2250         TAX                    USE MODE FOR INDEX
0BFE- BD ED 0C      2260         LDA MASK.ONE.TABLE,X   AND SET
0C01- 8D F9 0C      2270         STA MASK.ONE           MASKS
0C04- BD F1 0C      2280         LDA MASK.TWO.TABLE,X
0C07- 8D FA 0C      2290         STA MASK.TWO
0C0A- 60            2300         RTS
                    2310 *--------------------------------
                    2320 E.ZAP
0C0B- 98            2330         TYA                    START AT
0C0C- AA            2340         TAX                    CURSOR
0C0D- A9 A0         2350         LDA #SPACE
0C0F- 9D 57 07      2360 .1      STA EDIT.BUFFER,X
0C12- E8            2370         INX
0C13- E0 1E         2380         CPX #30                DONE?
0C15- 90 F8         2390         BCC .1
0C17- 60            2400         RTS
                    2410 *--------------------------------
                    2420 E.FIND
0C18- 20 A1 0C      2430         JSR GETKEY             GET SEARCH KEY
0C1B- 8D F6 0C      2440         STA SEARCH.KEY
0C1E- 98            2450 .1      TYA
0C1F- AA            2460         TAX
0C20- E8            2470 .2      INX                    START AT CURSOR+1
0C21- E0 1E         2480         CPX #30                END?
0C23- B0 1A         2490         BCS .3                 YES, NOT FOUND
0C25- BD 57 07      2500         LDA EDIT.BUFFER,X
0C28- CD F6 0C      2510         CMP SEARCH.KEY         MATCH?
0C2B- D0 F3         2520         BNE .2                 NO, NEXT X
0C2D- 8A            2530         TXA                    YES, MOVE CURSOR
0C2E- A8            2540         TAY
0C2F- 20 6E 0C      2550         JSR DISPLAY.EDIT.BUFFER
                    2560 *                              NEXT KEYPRESS
0C32- CD F6 0C      2570         CMP SEARCH.KEY         SAME CHARACTER?
0C35- F0 E7         2580         BEQ .1                 YES, FIND IT AGAIN
0C37- 68            2590         PLA                    NO, PULL A RETURN
0C38- 68            2600         PLA                    ADDRESS AND GO
0C39- AD F5 0C      2610         LDA CURRENT.CHAR
0C3C- 4C 5D 0B      2620         JMP REENTRY            PROCESS THIS KEY
                    2630
0C3F- 4C 3A FF      2640 .3      JMP MON.BELL           RETURN THROUGH BELL
                    2650 *--------------------------------
                    2660 E.RETURN
0C42- 20 5B 0C      2670         JSR MOVE.BUFFER.INTO.ARRAY
                    2680
                    2690 E.ESCAPE
0C45- 68            2700         PLA                    POP ONE RETURN
0C46- 68            2710         PLA                    ADDRESS AND RETURN
0C47- 60            2720         RTS                    TO ARRANGING
                    2730 *--------------------------------
                    2740 MOVE.FILE.INTO.BUFFER
0C48- AD 0C 0D      2750         LDA ACTIVE.ELEMENT     SET
0C4B- 20 E4 0A      2760         JSR POINT.TO.A         POINTER
0C4E- A0 03         2770         LDY #3
0C50- B1 00         2780 .1      LDA (POINTER),Y        MOVE
0C52- 99 54 07      2790         STA EDIT.BUFFER-3,Y    NAME
```

QUICKTRACE

relocatable program traces and displays the actual machine operations, *while* it is running without interfering with those operations. Look at these *FEATURES:*

***Single-Step** mode displays the last instruction, next instruction, registers, flags, stack contents, and six user-definable memory locations.*

***Trace** mode gives a running display of the Single-Step information and can be made to stop upon encountering any of nine user-definable conditions.*

***Background** mode permits tracing with no display until it is desired. Debugged routines run at near normal speed until one of the stopping conditions is met, which causes the program to return to Single-Step.*

***QUICKTRACE** allows changes to the stack, registers, stopping conditions, addresses to be displayed, and output destinations for all this information. All this can be done in Single-Step mode while running.*

***Two optional display formats** can show a sequence of operations at once. Usually, the information is given in four lines at the bottom of the screen.*

***QUICKTRACE** is completely transparent to the program being traced. It will not interfere with the stack, program, or I/O.*

***QUICKTRACE** is relocatable to any free part of memory. Its output can be sent to any slot or to the screen.*

***QUICKTRACE** is completely compatible with programs using Applesoft and Integer BASICs, graphics, and DOS. (Time dependent DOS operations can be bypassed.) It will display the graphics on the screen while **QUICKTRACE** is alive.*

***QUICKTRACE** is a beautiful way to show the incredibly complex sequence of operations that a computer goes through in executing a program*

QUICKTRACE *$50*

Is a trademark of Anthro-Digital, Inc.

Written by John Rogers

See these programs at participating Computerland and other fine computer stores.

Anthro - Digital Software, Inc.
P.O. Box 1385 Pittsfield, MA 01202

```
0C55- C8          2800         INY
0C56- C0 21       2810         CPY #$21
0C58- 90 F6       2820         BCC .1
0C5A- 60          2830         RTS
                  2840 *-------------------------------
                  2850 MOVE.BUFFER.INTO.ARRAY
0C5B- AD 0C 0D    2860         LDA ACTIVE.ELEMENT  MAKE
0C5E- 20 E4 0A    2870         JSR POINT.TO.A      POINTER
0C61- A0 03       2880         LDY #3
0C63- B9 54 07    2890 .1      LDA EDIT.BUFFER-3,Y MOVE
0C66- 91 00       2900         STA (POINTER),Y     NAME
0C68- C8          2910         INY
0C69- C0 21       2920         CPY #$21
0C6B- 90 F6       2930         BCC .1
0C6D- 60          2940         RTS
                  2950 *-------------------------------
                  2960 DISPLAY.EDIT.BUFFER
0C6E- A9 DD       2970         LDA #$DD            ]
0C70- 8D 56 07    2980         STA EDIT.BUFFER-1   LEFT END
0C73- A9 DB       2990         LDA #$DB            [
0C75- 8D 75 07    3000         STA EDIT.BUFFER+30  RIGHT END
0C78- A2 1D       3010         LDX #29
0C7A- A9 A0       3020 .1      LDA #SPACE
0C7C- 9D D7 06    3030         STA CONTROL.LINE,X REMOVE OLD CONTROL
0C7F- 9D D7 07    3040         STA CURSOR.LINE,X  BAR AND CURSOR
0C82- BD 57 07    3050         LDA EDIT.BUFFER,X
0C85- C9 A0       3060         CMP #$A0
0C87- B0 09       3070         BCS .2              CONTROL?
0C89- C9 80       3080         CMP #$80
0C8B- 90 05       3090         BCC .2
0C8D- A9 DF       3100         LDA #$DF     _      YES, PUT BAR
0C8F- 9D D7 06    3110         STA CONTROL.LINE,X
0C92- CA          3120 .2      DEX
0C93- 10 E5       3130         BPL .1
0C95- AD FB 0C    3140         LDA CURSOR          GET CURSOR,
0C98- 2D F9 0C    3150         AND MASK.ONE        CONDITION IT,
0C9B- 0D FA 0C    3160         ORA MASK.TWO
0C9E- 99 D7 07    3170         STA CURSOR.LINE,Y   AND SHOW IT
                  3180 *-------------------------------
0CA1- AD 00 C0    3190 GETKEY  LDA KEYBOARD
0CA4- 10 FB       3200         BPL GETKEY
0CA6- 8D 10 C0    3210         STA KEYSTROBE
0CA9- 8D F5 0C    3220         STA CURRENT.CHAR
0CAC- 60          3230         RTS
                  3240 *-------------------------------
                  3250 SEARCH.AND.PERFORM.NEXT
0CAD- E8          3260         INX                 NEXT ENTRY
0CAE- E8          3270         INX
0CAF- E8          3280         INX
                  3290
                  3300 SEARCH.AND.PERFORM
0CB0- BD C3 0C    3310         LDA EDIT.TABLE,X    GET VALUE FROM TABLE
0CB3- F0 05       3320         BEQ .1              NOT IN TABLE
0CB5- CD F5 0C    3330         CMP CURRENT.CHAR
0CB8- D0 F3       3340         BNE SEARCH.AND.PERFORM.NEXT
0CBA- BD C5 0C    3350 .1      LDA EDIT.TABLE+2,X  LOW BYTE OF ADDRESS
0CBD- 48          3360         PHA
0CBE- BD C4 0C    3370         LDA EDIT.TABLE+1,X  HIGH BYTE
0CC1- 48          3380         PHA
0CC2- 60          3390         RTS                 GO DO IT!
```

```
                    3400 *--------------------------------
                    3410 EDIT.TABLE
0CC3- 82 D7 0B      3420         .DA #$82,E.BEGINNING-1      ^B
0CC6- 84 C2 0B      3430         .DA #$84,E.DELETE-1         ^D
0CC9- 85 DA 0B      3440         .DA #$85,E.END-1            ^E
0CCC- 86 17 0C      3450         .DA #$86,E.FIND-1           ^F
0CCF- 88 AC 0B      3460         .DA #$88,E.LEFT.ARROW-1    <--
0CD2- 89 B7 0B      3470         .DA #$89,E.INSERT-1         ^I
0CD5- 8D 41 0C      3480         .DA #$8D,E.RETURN-1     RETURN
0CD8- 8F 9F 0B      3490         .DA #$8F,E.OVERRIDE-1       ^O
0CDB- 92 EC 0B      3500         .DA #$92,E.RESTORE-1        ^R
0CDE- 93 F1 0B      3510         .DA #$93,E.SET.MODE-1       ^S
0CE1- 95 B1 0B      3520         .DA #$95,E.RIGHT.ARROW-1   -->
0CE4- 9A 0A 0C      3530         .DA #$9A,E.ZAP-1            ^Z
0CE7- 9B 44 0C      3540         .DA #$9B,E.ESCAPE-1        ESC
0CEA- 00 39 FF      3550         .DA #$00,MON.BELL-1     OTHERS
                    3560 *--------------------------------
                    3570 MASK.ONE.TABLE
0CED- FF 3F 7F
0CF0- FF            3580         .DA #$FF,#$3F,#$7F,#$FF
                    3590
                    3600 MASK.TWO.TABLE
0CF1- 00 00 40
0CF4- 20            3610         .DA #$00,#$00,#$40,#$20
                    3620 *--------------------------------
0CF5-               3630 CURRENT.CHAR .BS 1
0CF6-               3640 SEARCH.KEY   .BS 1
0CF7-               3650 INPUT.MODE   .BS 1 (0 OR $FF)
0CF8-               3660 MASK.MODE    .BS 1 (0 TO 3)
0CF9-               3670 MASK.ONE     .BS 1 (FROM TABLE ABOVE
0CFA-               3680 MASK.TWO     .BS 1 (  "     "     "  )
0CFB-               3690 CURSOR       .BS 1 ($DE, $DC, OR $A2)
                    3700 *                  ( ^ ,  \ , OR  " )
```

Quickie No. 5.....Horst Schneider

To print a dashed line on the screen:

```
      JSR $FD9C     Print one dash
      JSR $FCA3     same character across screen
```

To print any character across screen:

```
      LDY #0
      LDA #$xx      xx = ASCII screen code for char
      JSR $FCA3
```

To print any character across most of screen:

```
      LDY #xx          xx = starting column
      LDA #$yy         yy = ASCII screen code for char
      JSR $FCA3
```

Adding Decimal Values from ASCII Strings...Bob Sander-Cederlof

The program below shows a nifty way to add two decimal values together and get the result as an ASCII string, without ever converting decimal to binary or binary to decimal.

The example shows two six-character values being added, but any length would work the same. For simplicity's sake I used a leading zero format, and allow no signs or decimal points. Fancier features can wait for more cerebral times.

The beautiful part is the way the 6502's carry flag works. On entering the add loop, I clear carry. Then I add a pair of digits, preserving the ASCII code. If the sum is more than "9" ($39), the CMP will leave carry set, prepared for subtracting 10 at line 1160. After subtracting 10, carry will be set (because the SBC caused no borrow). This carry then propagates to the next digit.

Strictly speaking, I should allow the sum to be one digit longer than the addend and augend strings, and store the final carry value there. Any reasonably useful version would also allow leading blanks and decimal points, be callable as an &-routine with string parameters, automatically handle non-aligned decimal points, and allow negative numbers. I'll try all these for next month.

```
                    1000 *SAVE S.STRING.ADD
                    1010 *--------------------------------
                    1020 *       STRING ADDITION
                    1030 *--------------------------------
0800- 30 30 30
0803- 31 38 39 1040 S1      .AS /000189/
0806- 30 30 37
0809- 30 33 30 1050 S2      .AS /007030/
                    1060 *--------------------------------
080C- 20 20 20
080F- 20 20 20 1070 S3      .AS /      /
                    1080 *--------------------------------
0812- A2 05    1090 ADD     LDX #5        6 DIGITS
0814- 18       1100         CLC           START WITH NO CARRY
0815- BD 00 08 1110 .1      LDA S1,X      NEXT DIGIT PAIR
0818- 29 0F    1120         AND #$0F      CHANGE ASCII TO BINARY CODE
081A- 7D 06 08 1130         ADC S2,X      RESULT IS IN ASCII AGAIN
081D- C9 3A    1140         CMP #$3A      UNLESS MORE THAN 9
081F- 90 02    1150         BCC .2        OKAY
0821- E9 0A    1160         SBC #10       NEED TO PROPAGATE CARRY
0823- 9D 0C 08 1170 .2      STA S3,X      SUM DIGIT
0826- CA       1180         DEX           MORE DIGITS?
0827- 10 EC    1190         BPL .1        YES
0829- 60       1200         RTS           NO, RETURN
```

Still More on Hardcore Magazine.....Bob Sander-Cederlof

I bought the latest, Vol. 1 No. 3, off the newsstand a few days ago. It is 72 pages, $3.50. I believe those 72 pages far surpass in usefulness the 600-odd pages of some familiar monthlies. A highlight for me was a complete assembly listing (in S-C format!) of HyperDOS, by John Bridges.

HyperDOS modifies the LOAD and BLOAD commands so that loading runs up to five times faster. This is the same improvement factor offered by a half dozen DOS-mods on the market, such as DOS Enhancer from S&H Software. (Of course, DOS Enhancer also speeds up SAVE and BSAVE, and include many other useful utilities with the package.)

If you are a nibble copier, you will be pleased with the listing of parameters for Locksmith and Nibbles Away II. As usual, there are a lot of hints on "how to unlock" those copy-protected disks: see "Controlling the I.O.B.", and "Boot Code Tracing".

Bev Haight (author of "Night Falls", among others) gives some excellent information on graphics, games, and even secrets to publishing. Bev describes, explains, and lists a new game called "Zyphyr Wars" for your pleasure and edification.

There is a lot more. Even an interview with Mike Markkula regarding Apple's position on software protection!

Issue number 4 promises to focus on graphics: novice-to-expert how-to's, complete graphic aid programs, tables, charts, reviews, etc.

The New "What's Where".....Bob Sander-Cederlof

Micro has doubled the size and tripled the value of their "What's Where in the Apple" book. There is now a 152-page double-column type-set 20-chapter text together with the previously published atlas and gazetteer. The new edition retails at $24.95 (our price $23).

If you already have the older edition, you only need the update, called "The Guide to What's Where", for $9.95 retail (our price ($9 even).

If you order books from us, remember to include enough for shipping.

HardCore Computing is a quarterly magazine.

Programming a Language Card.....Bill Morgan

Recently we've received a couple of questions about the exact meaning of all those $C08x addresses used to access a language (or RAM) card in slot 0. Here's a rundown of what memory cards are and how to use them.

A RAM card is a plug-in board containing an extra 16K (or more) of memory, which can be used instead of the language ROMs on your Apple motherboard. The $C08x addresses are switches that determine which memory will be used whenever you read or write an address from $D000-$FFFF. With the proper use of the switches on a 16K card, your Apple becomes a machine with 76K of memory! (That includes motherboard RAM, motherboard ROM, and the full RAM card.)

Here's a summary of the addresses and their functions:

Address	Read	Write	Bank
$C080	Card	Mother	2
$C081*	Mother	Card	2
$C082	Mother	Mother	2
$C083*	Card	Card	2
$C088	Card	Mother	1
$C089*	Mother	Card	1
$C08A	Mother	Mother	1
$C08B*	Card	Card	1

The stars indicate addresses which must be accessed twice to have effect (these are the ones that write-enable the card.)

These addresses are "soft switches", much like those for switching the screen display modes. To throw a switch, just use a LDA or any instruction that reads the location. From BASIC you can use a PEEK. STA or POKE also work with most RAM cards, but not all of them. Experiment with yours to see how it behaves. If you're writing a program for use on other people's Apples it's safest to stay with instructions that read the location.

The Bank column refers to the fact that a language card actually has 16K of memory, but the range from $D000 to $FFFF is only 12K. The other 4K ought to be $C000-$CFFF, but that's the area that Apple uses for special Input/Output functions. Therefore, there is an extra 4K "bank" which can be addressed at $D000-DFFF. Normally, only Bank 2 is used. If a program gets bigger than 12K it becomes necessary to use Bank 1, but that starts getting complicated. The best approach is to put routines or data in bank 1 that don't have to refer to anything in bank 2. You can then have the main code above $E000 decide which bank to use.

Some programs seem to use the motherboard and RAM card memories at the same time. Examples of this are ES-CAPE.LC and the programs that relocate DOS into the RAM card. Generally, these have a short "bridge" or "switcher" routine somewhere in the motherboard RAM. When the program in the RAM card needs to call a routine in the motherboard ROM, it actually calls the bridge. The bridge routine then throws the appropriate $C08x switches and calls the necessary ROM routine. When that routine finishes, the bridge then switches back to the RAM card and continues the program there.

Another thing to consider is whether the program in the RAM card needs the system monitor. If so, you need to make sure there is a copy of the monitor on the RAM card. Here's how to use the monitor to copy itself into a RAM card:

```
]CALL-151
*C081 C081
*F800<F800.FFFFM
```

That monitor move instruction looks like nonsense, but remember that the $C081 switch sets the computer to read from the motherboard and write to the RAM card.

The Book of Apple Software 1983

It's huge! Nearly 500 pages of insightful reviews and comparison charts, covering business, education, utilities, and games. The review of seven assemblers includes Merlin, Lisa 2.5, Tool Kit, LJK Edit 6502, MAE, S-C Assembler II (4.0) and S-C Macro Assembler. S-C Macro tied for first place with Merlin in the overall ratings, but surged ahead in the detail. Consider: not copy protected, typeset programmer reference card, cassette support, monitor and DOS commands without leaving assembler, FANTASTIC upgrade policy, RAM card optional, compressed source code, 32 character labels, and more.

Anyway, back to The Book....you owe it to yourself to consult therein before buying software. Even if the one you want to buy isn't in the book, you will get a broader perspective. I recommend it.

Seed Thoughts on Extensions.....Sanford Greenfarb

I am currently between computers. My 4 1/2 year old Apple died and I have ordered a Basis 108 to replace it. While waiting, I have been doing some thinking; I came to the conclusion that I can extend, by appropriate coding, either the monitor or Applesoft (or both) into the unused 4K bank of my 16K RAM card. That second 4K bank at $D000-DFFF is just sitting there, with nothing to do. In all the Apple mags I have seen no one approaches thi idea. Maybe they know something I don't, but as soon as my computer comes I am going to try it.

I suspect that I could insert code at $FF7A in the monitor to switch 4K banks and jump to $D000 for a modified character search subroutine. This way I could add more control characters and routines to the monitor. This would add features while keeping all the standard entry point address unchanged.

I don't know why no one has used this concept, or at least not publicly. I am offering this idea to you readers of Apple Assembly Line. I can't work on it until my new computer comes anyway, and you will probably think of a lot of good uses.

A Plug for some Neat New Products.....Richard Fabbri
Ridgefield, CT

Take a peek at BYTE Magazine, August 1982, Steve Ciarcia's article on the TMS9918-based Color graphics for Apple II. It has proved to be fantastic! You get 15 colors plus transparent, 32 sprite planes to overlay a 15-color hi-res of 256 dots by 192 lines. It works as advertised! Digital Dimensions (see BYTE, Nov 1982, page 352) advertises this as "E-Z Color" board, for $230. I have had one now for a month.

Two other neat new board for the Apple from Number Nine Computer Engineering Inc:

* a graphics board with 1024x1024 resolution; 256 colors from a palette of 4096; HARDWARE drawing of circles, arcs, rectangles and vectors; characters; area fill; light pen interface; $750 to $1090, depending on options.

* a processor card with 3.6 MHz 6502, 64K on-board high-speed RAM, transparent execution of all Apple II software, software-controlled speed for timed I/O operations; $745.

If you are interested: contact Number Nine at (203) 233-8134, or P.O.Box 1802, Hartford, CT 06144.

A Legible Phone Number for Computer Micro Works

Their ad last month was a little fuzzy around the area where the phone number was. The correct number is (305) 777-0268. George Beasley or his wife will take your order. This number is in Florida, where George is stationed with the Air Force.

I ordered one of their "Promette's". It is different than I thought, and better. Most such adapters will not work when a language card is in slot 0, because EPROM's are missing one of the enable lines the Apple uses. But the Promette has an active device inside which adds the extra enable line, so it works like you want it to. Another nice difference is that George's price is about 1/4 the normal price for these items.

Apple Assembly Line is published monthly by S-C SOFTWARE CORPORATION, P.O. Box 280300, Dallas, Texas 75228. Phone (214) 324-2050. Subscription rate is $15 per year in the USA, sent Bulk Mail; add $3 for First Class postage in USA, Canada, and Mexico; add $13 postage for other countries. Back issues are available for $1.50 each (other countries add $1 per back issue for postage).